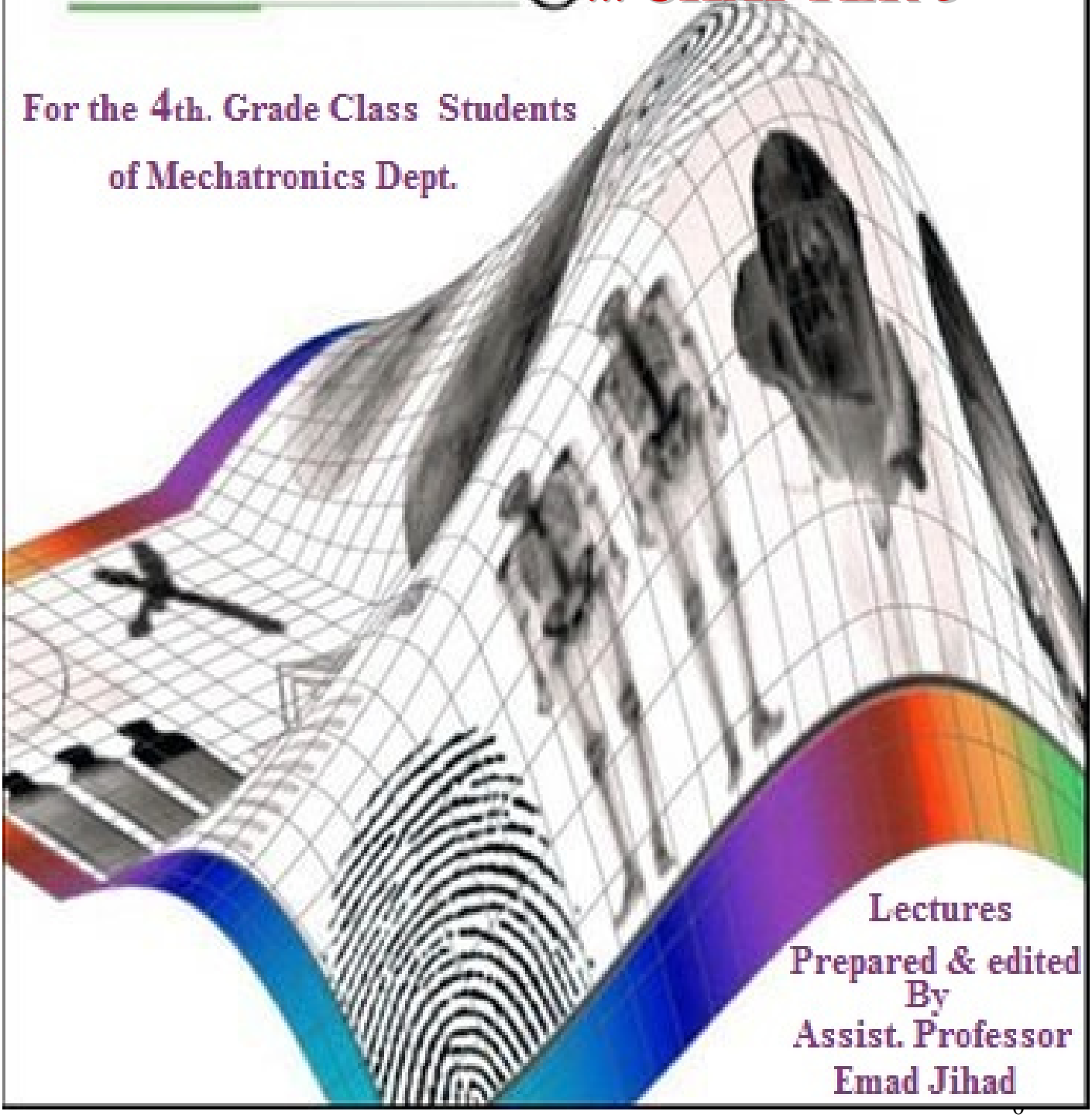
Digital Image Processing USING MATLAB

... CHAPTER 5

For the 4th. Grade Class Students

of Mechatronics Dept.



Lectures

Prepared & edited

By

Assist. Professor

Emad Jihad

5- Miscellaneous Image Processing operations

5.1 Digital image measurement units:

In a previous tutorial of spatial resolution (Chapter 1), we discussed the brief introduction of PPI, DPI, and LPI. Now we are formally going to discuss all of them.

5.1.1 Pixels per inch:

Pixel density or Pixels per inch is a measure of spatial resolution for different devices that includes Computers, tablets, and mobile phones.

The higher is the PPI, the higher is the quality. In order to more understand it, that how it calculated. Let's calculate the PPI of a mobile phone.

Example 5-1-1: Calculating pixels per inch (PPI) of Samsung galaxy S4:

The Samsung galaxy s4 has PPI or pixel density of 441. we will use ***Pythagoras theorem*** to calculate the diagonal resolution in pixels.

$$c = \sqrt{a^2 + b^2}$$

Where a and b are the height and width resolutions in pixel and

c is the diagonal resolution in pixels.

For Samsung galaxy s4, it is (1080 x 1920) pixels.

So putting those values in the equation gives the result

C = 2202.90717

Now we will calculate PPI

PPI = c / diagonal size in inches

The diagonal size in inches of Samsun galaxy s4 is 5.0 inches, which can be confirmed from anywhere.

PPI = 2202.90717/5.0

PPI = 440.58

PPI = 441 (approx)

That means that the **pixel density** of Samsung galaxy s4 is 441 PPI.

5.1.2 Dots per inch.

The dpi is often related to PPI, whereas there is a difference between these two. DPI or dots per inch is a measure of spatial resolution of printers. In case of printers, dpi means that how many dots of ink are printed per inch when an image gets printed out from the printer.

The higher is the dpi of the printer, the higher is the quality of the printed document or image on paper. Usually some of the laser printers have dpi of 300 and some have 600 or more.

5.1.3 Lines per inch

When dpi refers to dots per inch, liner per inch refers to lines of dots per inch. The resolution of halftone screen is measured in lines per inch.

5.2 Image size

The size of an image depends upon three things.

1. Number of rows
2. Number of columns
3. Number of bits per pixel

The formula for calculating the size is given below.

Size of an image = rows * cols * bpp

Example 5-2-1: Calculate a gray scale image size, assuming it has 1024 rows and it has 1024 columns. And since it is a gray scale image, so it has 256 different shades of gray or it has 8 bits per pixel. Then putting these values in the formula, we get:

= 1024 * 1024 * 8

= 8388608 bits.

Converting it into bytes = 8388608 / 8 = 1048576 bytes.

Converting into kilo bytes = 1048576 / 1024 = 1024kb.

Converting into Mega bytes = 1024 / 1024 = 1 Mb.

Calculating the mega pixels of the camera

Let's say we have an image of dimension: 2500 X 3192.

Its pixel resolution = 2500 * 3192 = 7982350 bytes.

Dividing it by 1 million = 7.9 = 8 mega pixel (approximately).

5.3 Histograms Introduction

Before discussing the use of Histograms in image processing, we will first look at what histogram is, how it is used and then an example of histograms to have more understanding of histogram.

5.3.1 Histograms:

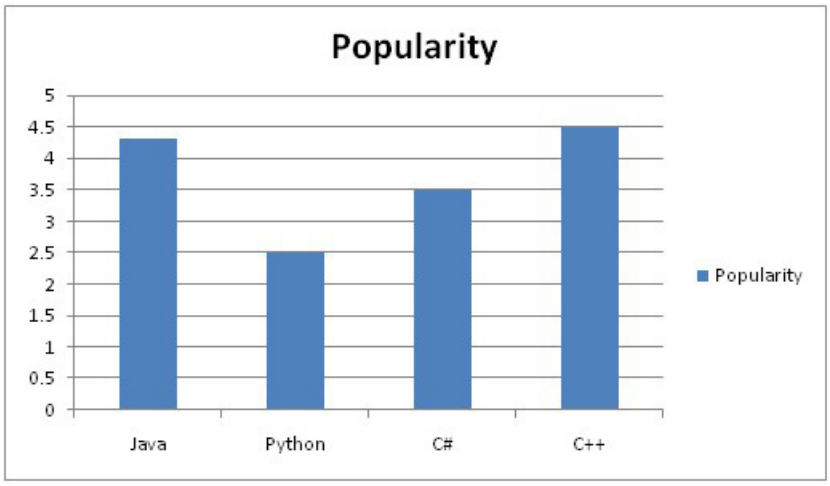
A histogram is a graph. A graph that shows frequency of anything. Usually histograms have bars that represent frequency of occurring of data in the whole data set. A Histogram has two axis the x axis and the y axis.

The x axis contains event whose frequency you have to count.

The y axis contains frequency.

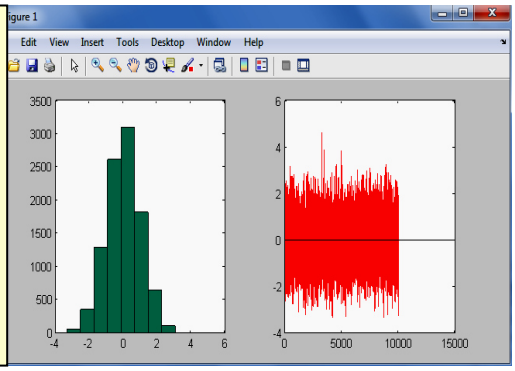
The different heights of bar shows different frequency of occurrence of data.

Usually a histogram looks like this.



(Note: sometimes, such a chart is called Bar chart for non-interval data, while Histogram is called for representing interval data; however, here we will consider the name "Histogram" instead of bar chart!). The following example illustrates this.

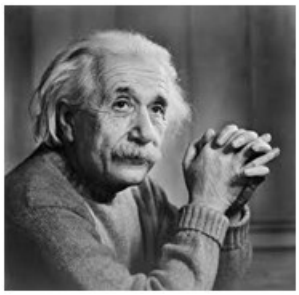
```
%**********************************
%* Histogram Example (Non-image)! *
%* Program name: HistEx01         *
%**********************************
clc; x = -4:.5:4;
y = randn(10000,1); % Normally distributed
pseudorandom numbers
subplot (1,2,1);hist(y); colormap summer
subplot (1,2,2);bar(y,'r')
input('Hit Enter key to Exit!');
close
```



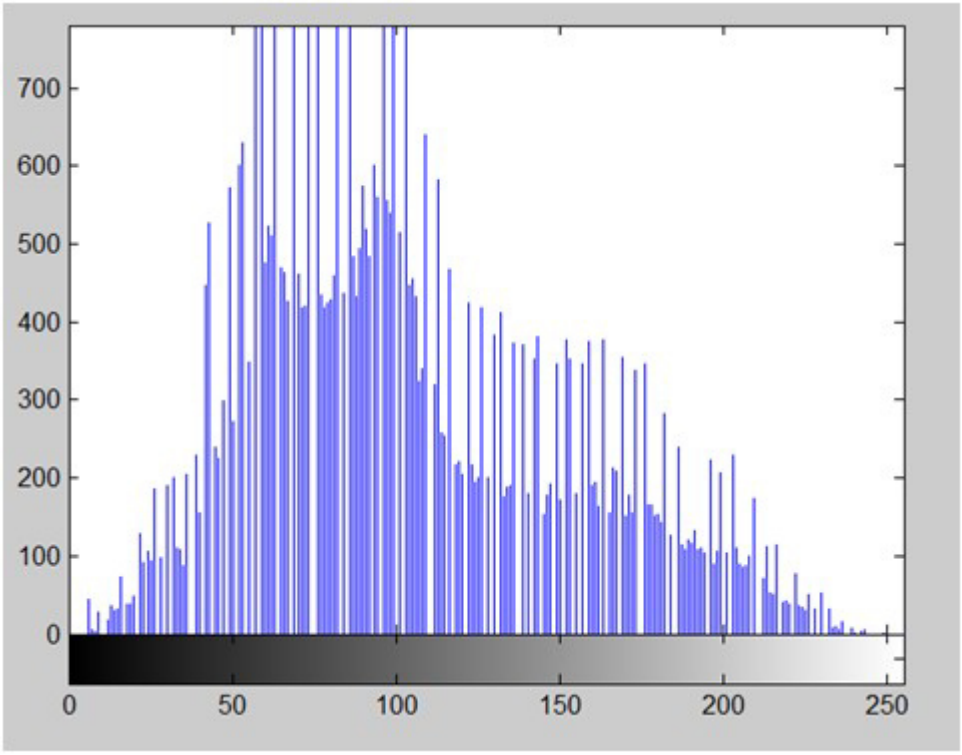
5.3.2 Histogram of an image

Histogram of an image, like other histograms also shows frequency. But an image histogram shows frequency of pixels intensity values. In an image histogram, the x axis shows the gray level intensities and the y axis shows the frequency of these intensities.

For example:



The histogram of the above picture of the Einstein would be something like this



The x axis of the histogram shows the range of pixel values. Since it's an 8 bpp image that means it has 256 levels of gray or shades of gray in it. That's why the range of x axis starts from 0 and end at 255 with a gap of 50. Whereas on the y axis, is the count of these intensities.

As you can see from the graph, that most of the bars that have high frequency lies in the first half portion which is the darker portion. That means that the image we have got is darker. And this can be proved from the image too.

5.3.3 Applications of Histograms:

Histograms have many uses in image processing. The first use as it has also been discussed above is the analysis of the image. We can predict about an image by just looking at its histogram. It's like looking an x ray of a bone of a body.

The second use of histogram is for brightness purposes. The histograms have wide application in image brightness. Not only in brightness, but histograms are also used in adjusting contrast of an image.

Another important use of histogram is to equalize an image.

And last but not the least; histogram has wide use in thresholding. This is mostly used in computer vision.

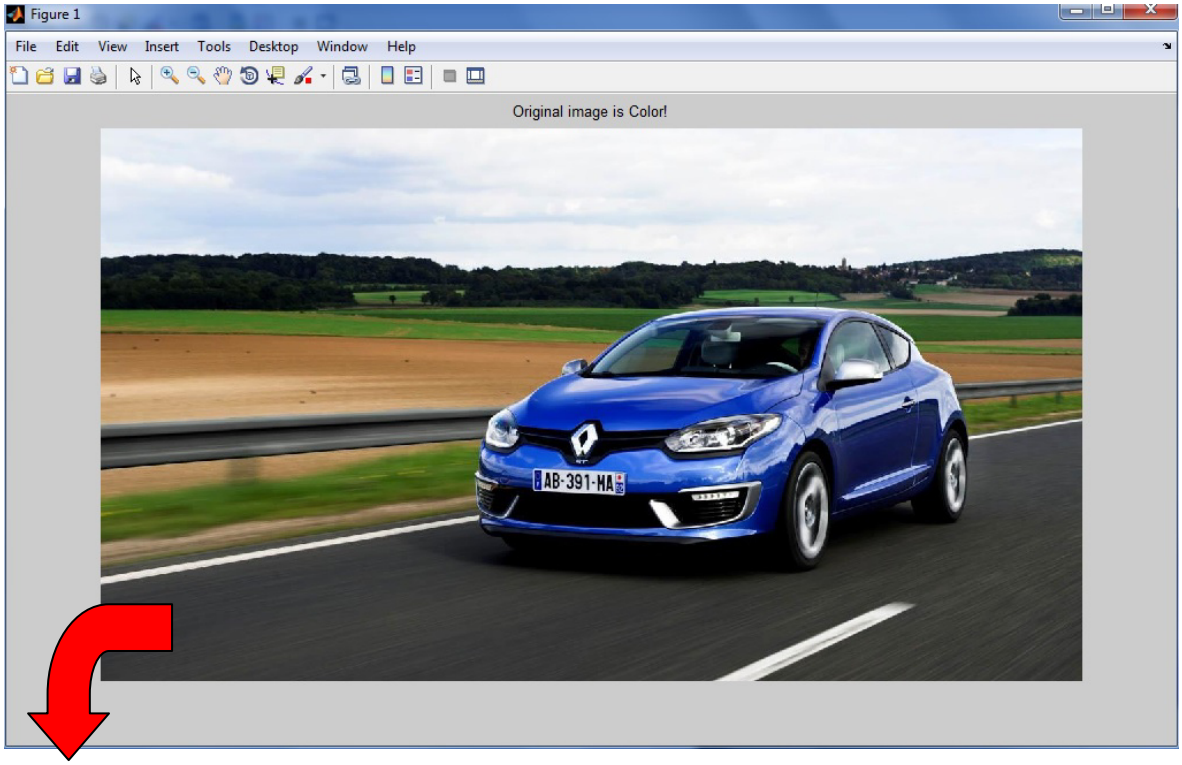
5.4 Image Histograms in Matlab:

To create a histogram for an image using the **`imhist`** function. An image histogram is a chart that shows the distribution of intensities in an indexed or **grayscale** image. The `imhist` function creates a histogram plot by defining *n* equally spaced bins, each representing a range of data values, and then calculating the number of pixels within each range. You can use the information in a histogram to choose an appropriate enhancement operation. For example, if an image histogram shows that the range of intensity values is small, you can use an intensity adjustment function to spread the values across a wider range.

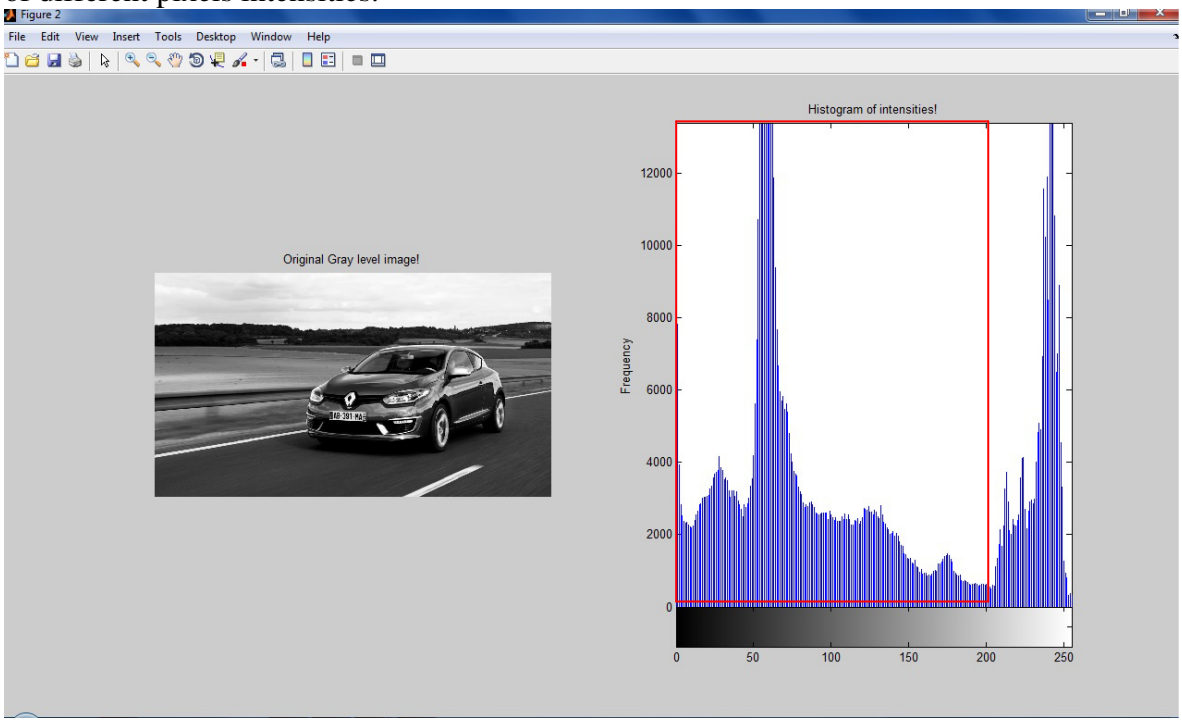
Example 5-4-1: Write Matlab scrip to read an image, if the image was Color then turn it into gray level then display it along with its histogram chart!

```
%*******************************
%* Gray Image Histogram Example*
%* Program name: ImageProEx25  *
%*******************************
clc
im = input('Enter image name: ','s');
K = imfinfo (im);
im = imread (im);
% Checking that the image is Not Color, otherwise converting it to gray
if K.BitDepth/8 == 3
   %im = imread (im);
   imshow (im) ; title('Original image is Color!');
   im =rgb2gray(im); figure;
   disp('Color image is Converted to Gray level!');
else
   %im = imread (im);
end
subplot(1,2,1); imshow(im); title ('Original Gray level image!');
subplot(1,2,2); imhist(im); title ('Histogram of intensities!');
ylabel('Frequency');

input('Hit Enter key to Exit!');
close all
```

If the image selected was Color, then it will be shown then converted to Gray scale:



Then the Histogram chart will be displayed along with gray image showing the frequencies of different pixels intensities.



Analyzing of this histogram found that frequencies of pixels < 200 gray intensity are over those > 200, thus the original image considered a little bit dark but not so much!

Hint: Chose different color or gray level images and test the histogram of each.

5.5 Improve Image Contrast

Sometimes, we get some low gray contrast images that can be checked using histograms as explained before, most of the pixels contrast intensity frequencies distribution will be in intensity range of the image that is rather narrow. The range does not cover the potential range of [0, 255], and is missing the high and low values necessary in good contrast.

In Matlab, there are several functions to improve image contrasts; one of these is **histeq** function. **Histogram equalization** spreads the intensity values over the full range of the image.

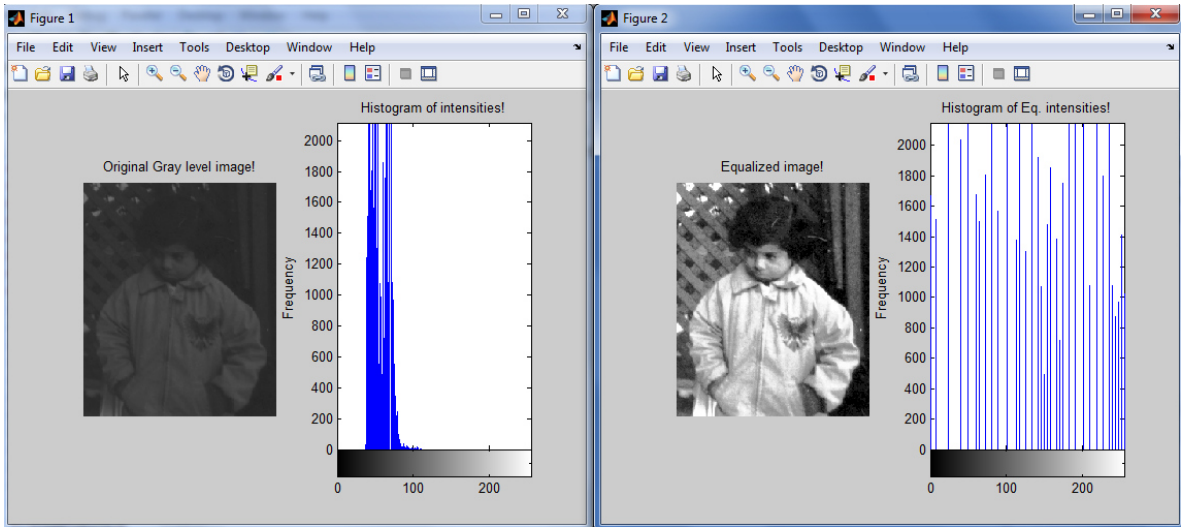
Example 5-5-1: Write Matlab script to read a gray level image, if it was color then convert it to gray then display it along with its histogram. Then improve the contrast using equalization and display the new improved image with histogram to compare between the two cases. The new image may be saved according to the user's request.

```
%*********************************************************
%* Gray Image Histogram and Histogram Equalization Example*
%* Program name: ImageProEx25a                            *
%*********************************************************
clc
im = input('Enter image name: ','s');
K = imfinfo (im);
im = imread (im);
% Checking that the image is Not Color, otherwise converting it to gray
if K.BitDepth/8 == 3
   %im = imread (im);
   imshow (im) ; title('Original image is Color!');
   im =rgb2gray(im); figure;
   disp('Color image is Converted to Gray level!');
else
   %im = imread (im);
end
subplot(1,2,1); imshow(im); title ('Original Gray level image!');
subplot(1,2,2); imhist(im); title ('Histogram of intensities!');
ylabel('Frequency');

input('Hit Enter key to Equalize the image histogram!');
figure;
ie = histeq(im);
subplot(1,2,1); imshow(ie); title ('Equalized image!');
subplot(1,2,2); imhist(ie); title ('Histogram of Eq. intensities!');
ylabel('Frequency');
wi = input('Would you like to save new equalized image (Y/N)? ','s');
if (wi == 'y') | (wi=='Y')
    in = input('Save as: ','s');
    imwrite (ie,in);
    disp ('New image file saved');
end
disp ('=========================');
input('Hit Enter key to Exit!')
close all
```

It's obvious that the Contrast of the new Equalized histogram (B) has been increased, thus we get a better image than the original one in (A).

A B

Comparison between image contrasts and histograms in both cases is seen below.



5.6 Arithmetic Operations on digital images

When applying arithmetic operations on digital images, which imply addition, subtraction, multiplication, and division, we must keep in mind to that the Matrices rules or special element-wise rules are applicable.

Using these arithmetic operations on ant types of images (providing that they have the same number of rows and columns as matrices) will yield different effects on the output images as those arithmetic operations were used.

Example 5-6-1: Write Matlab script to apply arithmetic operations on any two images with the same No. of rows and columns to examine different effects yield?

```
%*****************************************************
%* Arithmetic Operations on Image Matrices (Effects)  *
%* Program Name: ImageProEx28                         *
%*****************************************************
clc
disp ('Effects of Arithmetic operations on Image Matrices!');
disp ('=================================================');
disp ('Input Two Equal image Matrix size');
Im1 = input ('Enter image #1 file name: ','s');
Im2 = input ('Enter image #2 file name: ','s');
A = imread (Im1);
B = imread (Im2);
% Checking that both matrices have the same No. of rows & columns
if size(A) == size(B)

  C1 = A + B;
  C2 = A - B;
  C3 = B - A;
  C4 = A .* B;
  C5 = A .\ B;
  C6 = B .\ A;
  C7 = A./ B;
  C8 = B./ A;
  figure, imshow (A); title ('Image # 1: (A)');
  figure, imshow (B); title ('Image # 2: (B)');
  figure, imshow (C1); title ('Image A + B');
  figure, imshow (C2); title ('Image A - B');
  figure, imshow (C3); title ('Image B - A');
  figure, imshow (C4); title ('Image A .* B');
  figure, imshow (C5); title ('Image A .\ B');
  figure, imshow (C6); title ('Image B .\ A');
  figure, imshow (C7); title ('Image A ./ B');
  figure, imshow (C8); title ('Image B ./ A');
  subplot (2,5,1),imshow (A); title ('Image # 1: (A)')
  subplot (2,5,2),imshow (B); title ('Image # 2: (B)');
  subplot (2,5,3),imshow (C1); title ('Image A + B');
  subplot (2,5,4),imshow (C2); title ('Image A - B');
  subplot (2,5,5),imshow (C3); title ('Image B - A');
  subplot (2,5,6),imshow (C4); title ('Image A .* B');
  subplot (2,5,7),imshow (C5); title ('Image A .\ B');
  subplot (2,5,8),imshow (C6); title ('Image B .\ A');
  subplot (2,5,9), imshow(C7); title ('Image A ./ B');
  subplot (2,5,10),imshow(C8); title ('Image B ./ A');
  input ('When Finish, Hit Enter key!');
else
    beep;
    disp ('-------------------------------------------------')
    disp ('Process Aborted: Both images must have the same Dim!')
end
close all
```

After executing the program on two (nXm) image matrices, the following figures will illustrate the different affects occurred because of the arithmetic operations used.

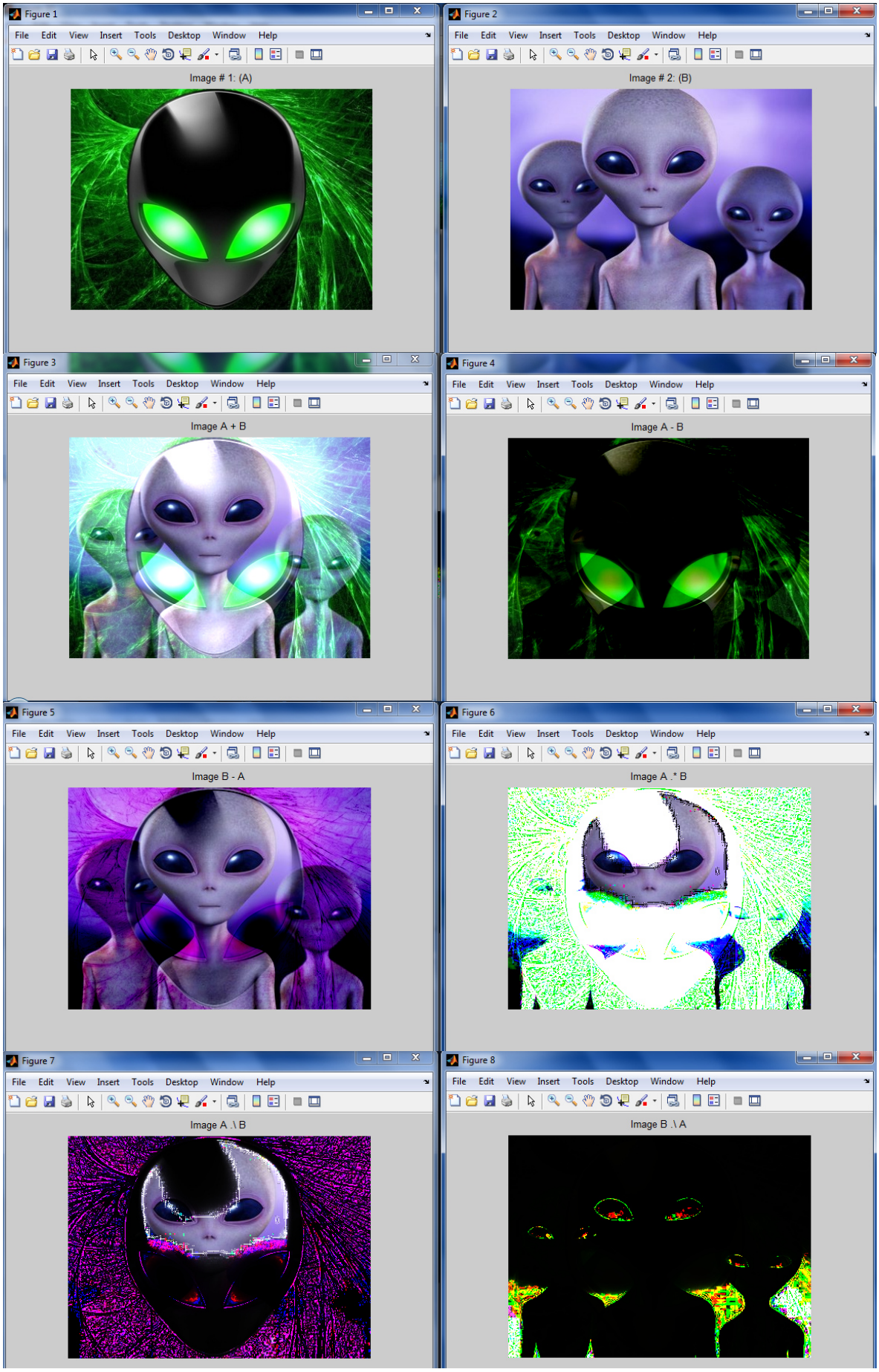
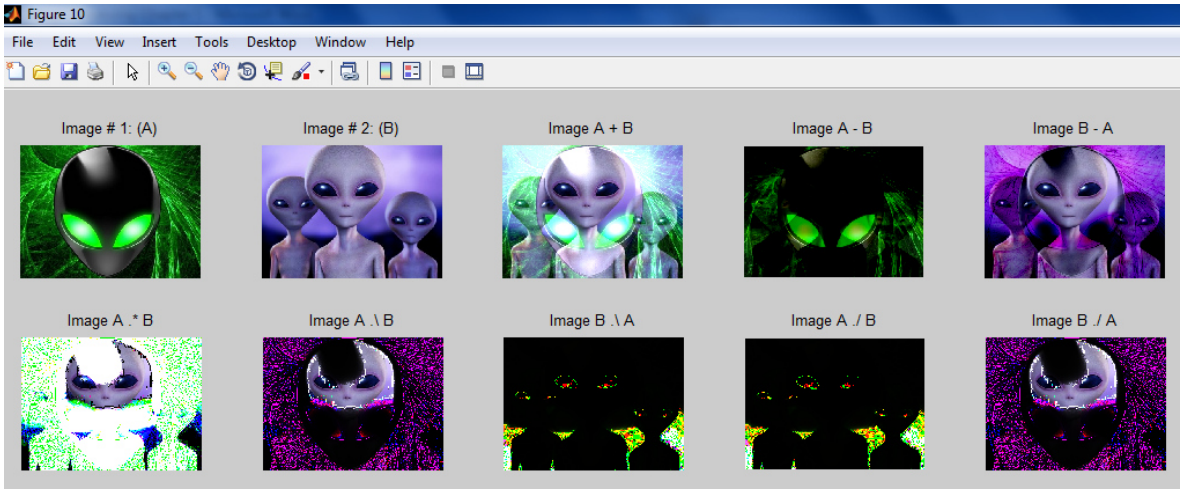


Figure 1
File Edit View Insert Tools Desktop Window Help
Image # 1: (A)
Figure 2
File Edit View Insert Tools Desktop Window Help
Image # 2: (B)
Figure 3
File Edit View Insert Tools Desktop Window Help
Image A + B
Figure 4
File Edit View Insert Tools Desktop Window Help
Image A - B
Figure 5
File Edit View Insert Tools Desktop Window Help
Image B - A
Figure 6
File Edit View Insert Tools Desktop Window Help
Image A .* B
Figure 7
File Edit View Insert Tools Desktop Window Help
Image A .\ B
Figure 8
File Edit View Insert Tools Desktop Window Help
Image B .\ A



Analyzing the different effects occurred, some can conclude the following:

1- Arithmetic Operations A+B and B+A give the same effect.
2- Element-wise A.* B and B.*A give the same result effect.
3- Element-wise A.\ B and B./ A give the same effect.
4- Also B.\A and A./B give the same result effect.
5- Addition operation always gives more bright contrast image.
6- Subtraction gives less contrast (dark) image.
7- Multiplication operation gives brighter pixels image.
8- Division operation gives darker pixels image.

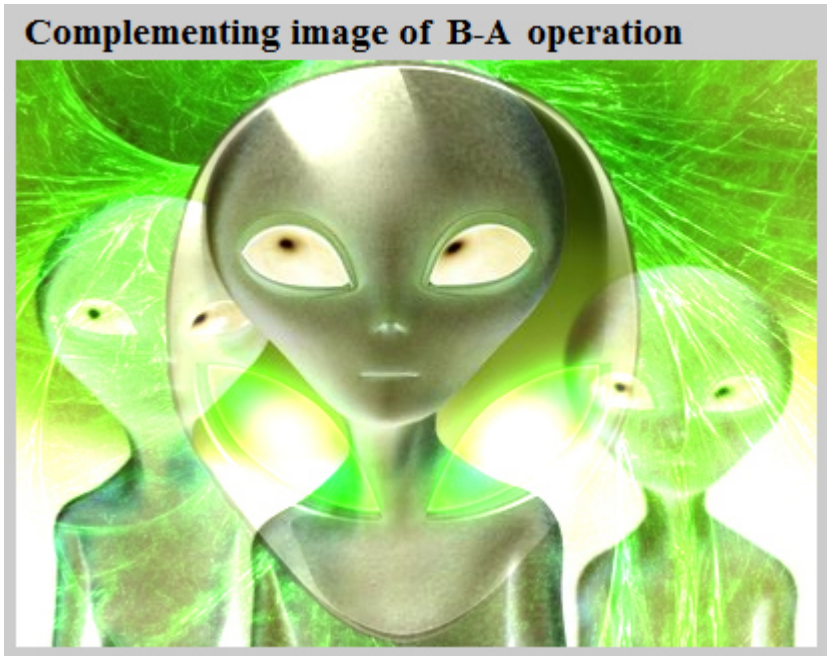
(Note: In case of any two Matrices such as A and B, the division operator mentioned above like in any result variable such as X = B / A is a solution to X * A = B, … While if using: X = A \ B is a solution to A * X = B.

This means: $A/B = A * B^{-1}$, while $A\backslash B = A^{-1} * B$, and the same is applied in case of $B/A = B * A^{-1}$, and $B\backslash A = B^{-1} * A$).

We may also make other image processing effects on any output image. For instant, taking the **complement** of C3 = (A-B) from the program :

```
figure; Neg = imcomplement (C3); imshow(Neg)
```

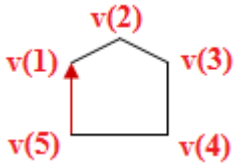
Gives:



5.7 Region Of Interest (ROI)

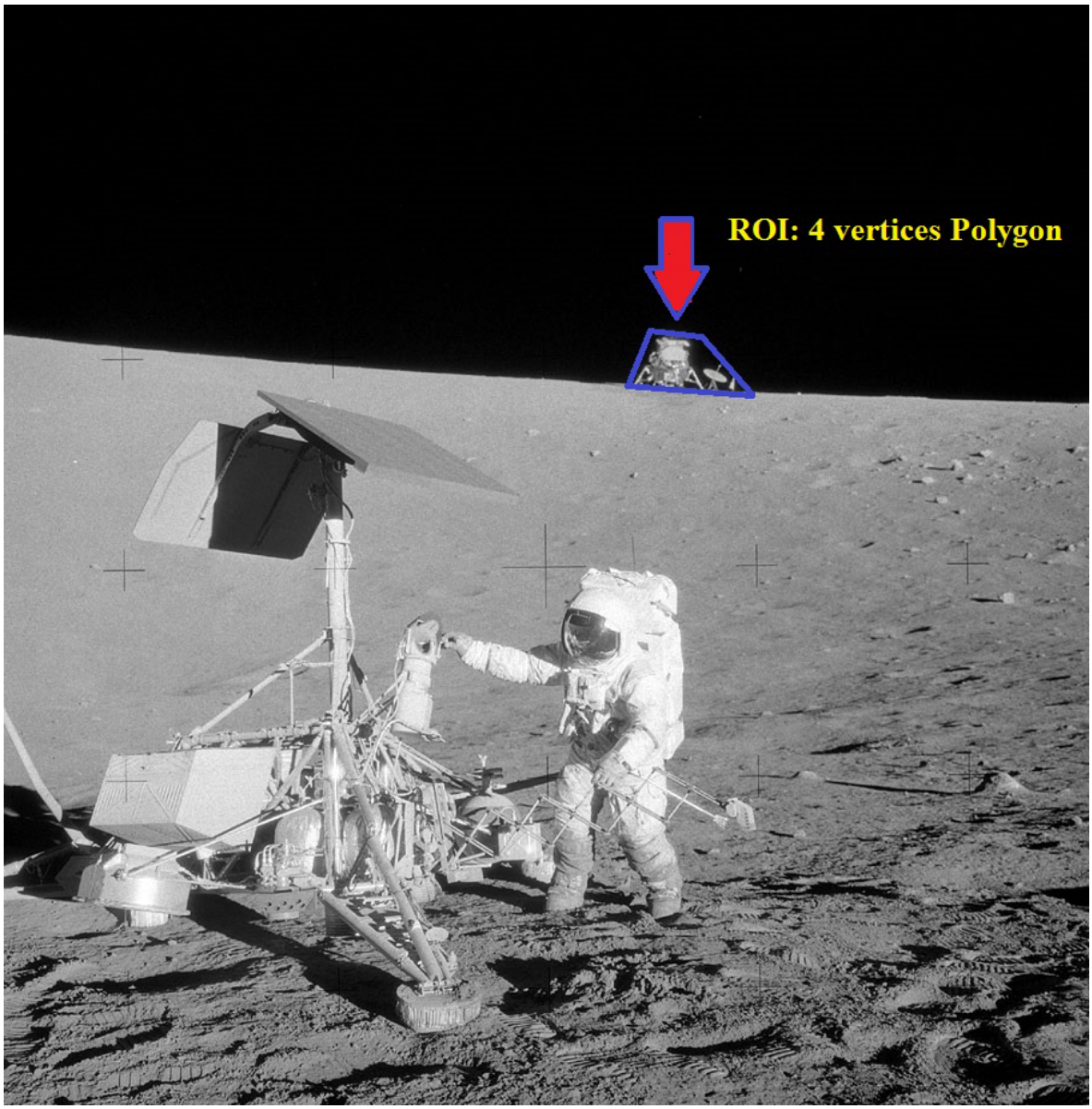
Region of Interest (ROI) is a subarea or a sub image defined from the original image using a **polygon** area. Within this polygon several processes may take place. This technique is very important in many fields that use image processing, for instant, in medical imaging, the boundaries of a tumor may be defined on an image or in a volume, for the purpose of measuring its size. Another use of ROI is to fill ROI area, such as to remove registration number from a vehicle's plate for example …and so forth…

ROI area represented by a polygon with vertices number v(i) = > (3), for example a 5 vertices polygon:



ROI has many implementations in Matlab, Here we will show only two of them.

The Following figure shows a ROI to be considered:



5.7.1 ROI Fill

Use **roifill** to fill in a specified region of interest (ROI) polygon in a grayscale image. roifill smoothly interpolates inward from the pixel values on the boundary of the polygon by solving Laplace's equation. The boundary pixels are not modified. roifill can be used, for example, to erase objects in an image of gray scale. Some Syntaxes of this function are:

J = roifill ……..1
J = roifill (I) ……..2
J = roifill (I, c, r) …….3

Where, **J** is the output Image with a ROI filled. **I** is the Input Gray scale image, **c** and **r** are equal-length vectors containing the row-column coordinates of polygon vertices.

Description

Use roifill to fill in a specified region of interest (ROI) polygon in a grayscale image. roifill smoothly interpolates inward from the pixel values on the boundary of the polygon by solving Laplace's equation. The boundary pixels are not modified. roifill can be used, for example, to erase objects in an image.

J = roifill (Syntax (1)), creates an interactive polygon tool, associated with the image displayed in the current figure, called the target image. You use the mouse to define the ROI.

J = roifill(I) (Syntax (2)), displays the image I and creates an interactive polygon tool associated with the image.

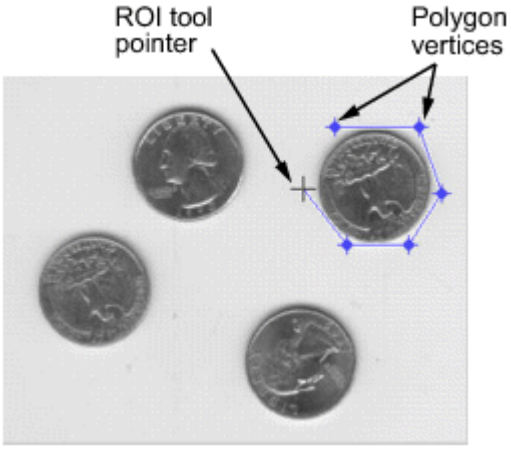
J = roifill(I, c, r) (Syntax (3)), fills in the polygon specified by c and r, which are equal-length vectors containing the row-column coordinates of the pixels on vertices of the polygon. The kth vertex is the pixel (r(k),c(k)).

Interactive Behavior

When you call `roifill` with an interactive syntax, the pointer changes to a cross hairs shape when you move it over the target image. Using the mouse, you specify a region-of-interest by selecting vertices of a polygon. You can change the size or shape of the polygon using the mouse. The following figure illustrates a polygon defined by multiple vertices. For more information about all the interactive capabilities of `roifill`, see the table that follows.

Interactive Behavior	Description
Closing the polygon. (Completing the region-of-interest.)	Use any of the following mechanisms: ▪ Move the pointer over the initial vertex of the polygon that you selected. The shape changes to a circle. Click either mouse button. ▪ Double-click the left mouse button. This action creates a vertex at the point under the mouse and draws a straight line connecting this vertex with the initial vertex. ▪ Click the right mouse button. This action draws a line connecting the last vertex selected with the initial vertex; it does not create a new vertex.
Deleting the polygon	Press **Backspace**, **Escape** or **Delete**, or right-click inside the region and select **Cancel** from the context menu. Note: If you delete the ROI, the function returns empty values.
Moving the polygon	Move the pointer inside the region. The pointer changes to a fleur. Click and drag the mouse to move the polygon.
Changing the color of the polygon	Move the pointer inside the region. Right-click and select **Set color** from the context menu.
Adding a new vertex.	Move the pointer over an edge of the polygon and press the **A** key. The shape of the pointer changes. Click the left mouse button to create a new vertex at that position on the line.
Moving a vertex. (Reshaping the region-of-interest.)	Move the pointer over a vertex. The pointer changes to a circle. Click and drag the vertex to its new position.
Deleting a vertex.	Move the pointer over a vertex. The pointer changes to a circle. Right-click and select **Delete Vertex** from the context menu. This action deletes the vertex and adjusts the shape of the polygon, drawing a new straight line between the two vertices that were neighbors of the deleted vertex.
Retrieving the coordinates of the vertices	Move the pointer inside the region. Right-click and select **Copy position** from the context menu to copy the current position to the Clipboard. Position is an *n*-by-2 array containing the *x*- and *y*-coordinates of each vertex, where *n* is the number of vertices you selected.

When using the interactive mode to define the polygon, it will be such as shown below:



In this example, 6 polygon vertices are defined By the mouse according to the rules mentioned Above on the table.

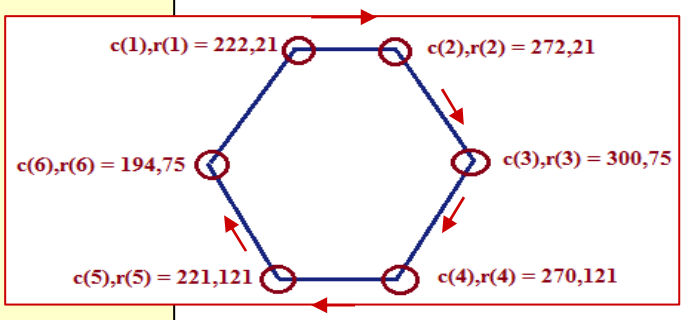
In the following example, 3rd. Syntax is used to manually enter the polygon's (ROI) vertices and erase that area using image background gray level color.

Example 5-7-1-1: Write Matlab script to erase the ROI manually by defining an n vertices polygon for any gray level image (Check the image for being a gray level, Otherwise convert it to gray level image)?

```
%*************************************************************************
%* Fill in specified region of interest (ROI) polygon in grayscale image*
%* Program name: ImageProEx32                                            *
%*************************************************************************
clc
close all
A = input ('Enter Gray level image name: ','s');
I =imread(A);
K = imfinfo (A);  % Get image information

% Checking that the image is Not Color, otherwise converting it to gray
if K.BitDepth/8 == 3
    disp('Color image is Converted to Gray level!');
    I =rgb2gray(I);
else

end
imshow(I), title('Source image');
% Hint: use the following Polygon Vertex coordinates for Coins ex. only!
% Columns: 222   272   300   270   221   194
% Rows   :  21    21    75   121   121    75

N = input ('Enter No. of Polygon verticies: ');
for i=1:N
   c(i) = input('Column: ');
   r(i) = input('Row   : ');
end

J = roifill(I,c,r);

figure, imshow(J), title('ROI Removed1');
input ('Hit Enter key to close!');
close all
```

Notice below that the user should enter the polygon ROI vertices in sequence:

```
Enter Gray level image name: coins.jpg
Color image is Converted to Gray level!
Enter No. of Polygon vertices: 6
Column: 222
Row   : 21
Column: 272
Row   : 21
Column: 300
Row   : 75
Column: 270
Row   : 121
Column: 221
Row   : 121
Column: 194
Row   : 75
```



5.7.2 ROI Mask

A Mask is a Polygon area that can be used later for filtering gray level images.
In Matlab, this could be done interactively or by entering the polygon row and column vertices as explained before in section 5.7.1.

roipoly is a Matlab function to Specify polygonal region of interest (ROI).
Syntax:

BW = roipoly ……(1)
BW = roipoly(I) ……(2)
BW = roipoly(I, c, r) ……(3)
BW = roipoly(x, y, I, xi, yi) ……(4)
[BW, xi, yi] = roipoly(...) ……(5)
[x, y, BW, xi, yi] = roipoly(...) ……(6)

BW is the ROI mask of the image.
Rules of section 5.7.1 are also applied for syntax (1) and (2) for interactive tools.
While,

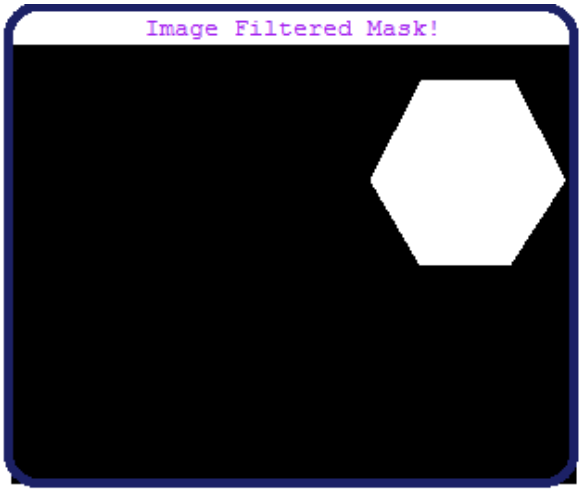
BW = roipoly(I, c, r) returns the ROI specified by the polygon described by vectors c and r, which specify the column and row indices of each vertex, respectively. c and r must be the same size.

BW = roipoly(x, y, I, xi, yi) uses the vectors x and y to establish a nondefault spatial coordinate system. xi and yi are equal-length vectors that specify polygon vertices as locations in this coordinate system.

[BW, xi, yi] = roipoly(...) returns the x- and y-coordinates of the polygon vertices in xi and yi.

Example 5-7-2-1: Write Matlab script to create a mask image, BW, the same size as the input image, I.

Solution:
Here, we will specify the polygon ROI mask thus to get:



The programs script may look like the following:

```
%***********************************
%* Mask Filtering for Gray images! *
%* Program name: ImageProEx29        *
%***********************************
clc
close all
t = 'Y';
disp ('This is an example of Gray image ROI Mask filter!');
disp ('=============================================================');
while (t=='Y') || (t=='y')
    clc
    A = input ('Enter Gray level image file name: ','s');
    I = imread(A);
    I =rgb2gray(I);
    % Hint: use the following Polygon Vertex coordinates for Coins
ex. only!
    % Columns: 222   272   300   270   221   194
    % Rows   :  21    21    75   121   121    75

    N = input ('Enter No. of Polygon vertices: ');
   for i=1:N
     c(i) = input('Column: ');
     r(i) = input('Row   : ');
   end
   BW = roipoly(I,c,r);
   figure, imshow(I),title ('Original image');
   figure, imshow(BW),title ('Image Filtered Mask!');
   t = input ('Another try (Y/N)? ','s');
   close all

end
```

5.7.3 ROI Filter

roifilt2 Filter region of interest (ROI) in image
Syntax:

J = roifilt2(h, I, BW)

J = roifilt2(h, I, BW) filters the data in I with the two-dimensional linear filter h. BW is a binary image the same size as I that defines an ROI used as a mask for filtering. roifilt2 returns an image that consists of filtered values for pixels in locations where BW contains 1's, and unfiltered values for pixels in locations where BW contains 0's. For this syntax, roifilt2 calls filter2 to implement the filter.

5.7.4 Creating a 2D Filter

We can create several Filters to be used with or without ROI of an image. These filters depend on Mathematical Algorithms as we will show later on.
fspecial is used to define and create the required filter as in the following:

Syntax

h = fspecial(type)
h = fspecial(type, parameters)

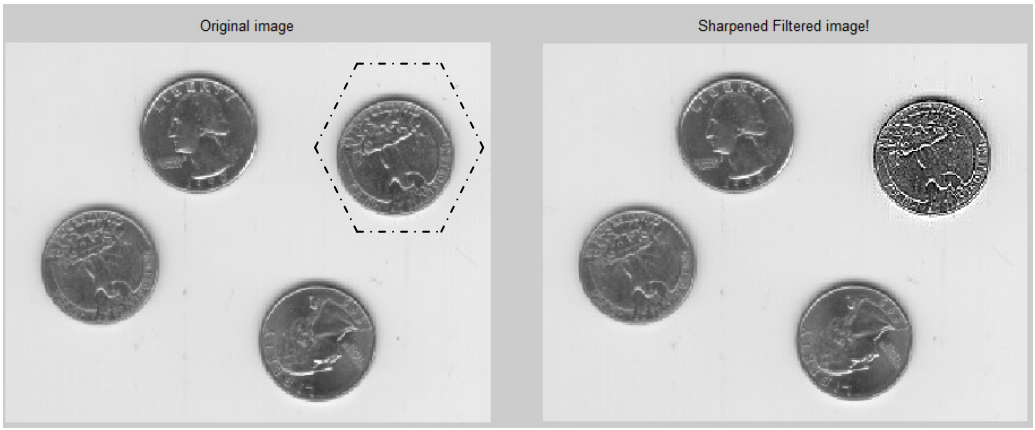
h = fspecial(type) creates a two-dimensional filter h of the specified type. fspecial returns h as a correlation kernel, which is the appropriate form to use with **imfilter**. type is a string having one of these values.

Value	Description
'average'	Averaging filter
'disk'	Circular averaging filter (pillbox)
'gaussian'	Gaussian lowpass filter
'laplacian'	Approximates the two-dimensional Laplacian operator
'log'	Laplacian of Gaussian filter
'motion'	Approximates the linear motion of a camera
'prewitt'	Prewitt horizontal edge-emphasizing filter
'sobel'	Sobel horizontal edge-emphasizing filter
'unsharp'	Unsharp contrast enhancement filter

Note: Do not be confused by the name of this filter: an **unsharp** filter is an operator used to Sharpen images. The name comes from a publishing industry process in which an image is sharpened by subtracting a blurred (unsharp) version of the image from itself.

In This stage we will not elaborate with these types. However, example below will show how to use the filter created with ROI of an image.

Example 5-7-4-1: Write Matlab script to use a filter for Sharpening a RIO of a gray scale image?



To accomplish this, the script used could be something like the following one:

```
%**********************************
%* Filtering ROI for Gray images!  *
%* Program name: ImagePro32a       *
%**********************************

close all
t = 'Y';
disp ('This is an example of Gray image ROI (Sharp) filter!');
disp ('=============================================');
while (t=='Y') || (t=='y')
    clc
    A = input ('Enter Gray level image file name: ','s');
    I = imread(A);
    I =rgb2gray(I);
    % Hint: use the following Polygon Vertex coordinates for
Coins ex. only!
    % Columns: 222   272   300   270   221   194
    % Rows   :  21    21    75   121   121    75

    N = input ('Enter No. of Polygon vertices: ');
   for i=1:N
     c(i) = input('Column: ');
     r(i) = input('Row   : ');
   end
   BW = roipoly(I,c,r);
   H = fspecial('unsharp');
   J = roifilt2(H,I,BW);
   subplot (1,2,1); imshow(I),title ('Original image');
   subplot (1,2,2); imshow(J),title ('Sharpened Filtered
image!');
   t = input ('Another try (Y/N)? ','s');
   close all

end
```

5.8 Activating Video Webcam

To conclude the lectures of image processing, let's take a look on how to activate any video camera connected to your PC (this include the built in webcam or any external device).
A video is actually a long stream of images that can be displayed as a motion video.
In Matlab, we will need first to specify any video device (such as a webcam) attached to the PC. This is done by displaying the information about this issue from windows by using Matlab's **imaqhwinf** function, which returns out, a structure that contains information about the image acquisition adaptors available on the system.
An adaptor is the interface between MATLAB and the image acquisition devices connected to the system. The adaptor's main purpose is to pass information between MATLAB and an image acquisition device via its driver.
Syntax:
Imaqhwinf…… (It's better to define a variable name to hold devices information),
Such as, info = imaqhwinf.

If we know the adapter name, then we can put it with the function:
imaqhwinf (adaptorname)
for instance, info = imaqhwinf ('winvideo').

Second, after we knew how many devices are attached to the system, we may show the device information.
Third, use **`videoinput`** function to activate the video, then **`preview`** to open video screen.
Example 5-8-1: Write necessary Matlab script to detect video devices attached to a PC system, display information about, then preview video via webcam on screen.
Solution:

```
%**********************************
%* Using Video (webcam) in Matlab *
%* Program name: ImageProEx40     *
%**********************************
clc
disp ('Getting computer video resources information!')
disp ('=============================================')
info = imaqhwinfo('winvideo')
info.DeviceInfo(1)  % Capital D and I to be used in DeviceInfo!
vid = videoinput('winvideo',1);
preview(vid);
input ('When Ready, Hit Enter key to Close Video Preview!');
closepreview;
```

Image Processing Lectures
Prepared, Programmed and edited by: Assist. Professor
Emad Jihad, 2014-2015
Mechatronics Dept.
Eng. Technical College-Baghdad
Mid. Technical University
IRAQ.

Wishing you All the Best!